Refine Search

Search Results -

Terms	Documents
PREPROCESSOR AND DIRECTIVE AND COLOR	20

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L1

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, December 03, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L1	PREPROCESSOR AND DIRECTIVE AND COLOR	20	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 20 of 20 returned.

☐ 1. Document ID: US 6804662 B1

L1: Entry 1 of 20 File: USPT Oct 12, 2004

US-PAT-NO: 6804662
DOCUMENT-IDENTIFIER: US 6804662 B1

TITLE: Method and apparatus for query and analysis

DATE-ISSUED: October 12, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Annau; Thomas M.	San Francisco	CA		
Sill; Joseph	San Francisco	CA		

US-CL-CURRENT: 707/2; 707/4

ABSTRACT:

A data handling method combines search capabilities with analytical functionality. The invention provides advantages when dealing with structured documents (such as electronic catalogs, XML documents, text documents, HTML documents, Internet documents, etc.) and other data stored in a computer system. Various embodiments include simplified ways to express search/analysis requests of a data set and also to express results to such requests.

26 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 2. Document ID: US 6732296 B1

L1: Entry 2 of 20 File: USPT May 4, 2004

US-PAT-NO: 6732296
DOCUMENT-IDENTIFIER: US 6732296 B1

TITLE: Object oriented scaleable test executive

DATE-ISSUED: May 4, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cherny; Jeffrey G.	Chesterland	OH		
Gehring; David A.	Mentor	OH		
Lupica; Robert L.	Willoughby	OH		
Wojcik; James A.	Parma	OH		

US-CL-CURRENT: 714/32; 717/107

ABSTRACT:

The present invention provides for a test system having a test executive software system for performing tests on units under test. The test executive software system includes a test kernel component that provides control through a generic interface to the test executive software. Test components, instrument components, support objects and a test system interface component are communicatively coupled to the test kernel component. The instrument components can be written as a dynamically linked library (DLL) file so that the instrument component can be broken into basic functional modules associated with the particular instrument type. Each instrument component supports operation in both live mode and virtual mode, so that testing can be performed in both normal mode and simulation mode. Virtual mode allows instruments to be inserted and removed without impacting test applications that do not utilize them, thereby reducing tester downtime.

22 Claims, 19 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 15

☐ 3. Document ID: US 6691301 B2

L1: Entry 3 of 20 File: USPT Feb 10, 2004

US-PAT-NO: 6691301
DOCUMENT-IDENTIFIER: US 6691301 B2

TITLE: System, method and article of manufacture for signal constructs in a programming language capable of programming hardware architectures

DATE-ISSUED: February 10, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bowen; Matt	Littlemore			GB

US-CL-CURRENT: 717/114; 712/15, 716/16

ABSTRACT:

A system, method and article of manufacture are provided for using a dynamic object in a programming language. In general, an object is defined with an associated

first value and second value. The first value is used in association with the object during a predetermined clock cycle. The second value is used in association with the object before or after the predetermined clock cycle.

18 Claims, 129 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 117

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 4. Document ID: US 6640145 B2

L1: Entry 4 of 20 File: USPT Oct 28, 2003

US-PAT-NO: 6640145
DOCUMENT-IDENTIFIER: US 6640145 B2

TITLE: Media recording device with packet data interface

DATE-ISSUED: October 28, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven	West Harrison	NY	10604	
Hoffberg-Borghesani; Linda	Acton	MA	01720	

US-CL-CURRENT: 700/83; 700/17, 700/19, 700/23, 704/200, 704/201, 704/7, 709/200, 709/201, 709/202

ABSTRACT:

An intelligent media device, comprising a packet data communications interface; a media communication interface for receiving audio and/or video data; a digital memory for persistently storing received audio and/or video data; and an intelligent server for generating a virtual interface for controlling the media communication interface and the digital memory through said packet data communications interface. The intelligent server may be adaptive. A variety of devices may be interfaced through the packet data communications interface, including telephony, imaging, videoconferencing, security, alarm, environmental control, vehicular, illumination system, domestic appliance; fluid and handling systems, as well as consumer electronic devices. A digital rights manager for enforcing a set of externally supplied restrictions associated with the received audio and/or video data may be incorporated, with a cryptographic processor for selectively cryptoprocessing audio and/or video data in dependence on said rights manager being provided to limit access to the audio and/or video data content.

23 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 28

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 5. Document ID: US 6501950 B1

L1: Entry 5 of 20 File: USPT Dec 31, 2002

US-PAT-NO: 6501950
DOCUMENT-IDENTIFIER: US 6501950 B1

TITLE: Systems and methods for monitoring data signals on a communications network

DATE-ISSUED: December 31, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Smith; Thomas C.	Roswell	GA		
Morton; John W.	Cumming	GA		
Seymour; Jefferey T.	Douglasville	GA		
Visser; Gregory S.	Dunwoody	GA		
Henseler; David L.	Norcross	GA		

US-CL-CURRENT: 455/423; 455/412.1, 455/514

ABSTRACT:

A system and method capture data from an SS7 network and pair each invoke message with its corresponding response message. Each message is stored in a daily file as well as in a table dedicated for that particular message type. A separate file is also maintained on each MIN and holds the most current registration information for each MIN. The system and method support four different types of queries: a MIN/ESN query, an active roamer query, a switch-to-switch query, and a transaction statistics query. The MIN/ESN query provides roaming activity on a specific MIN or ESN while the active roamer query provides information on all roamers in a serving market. The switch-to-switch query reveals messaging at one switch or between two switches. The statistics query provides data to a provider on all of its subscribers roaming in foreign networks or on all phones within its own network. The system has a Graphical User Interface (GUI) for displaying information on each message in an easy and convenient manner. The details of a record are displayed in a tabular format with each tab holding data for a group of parameters.

6 Claims, 63 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 63

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 6. Document ID: US 6487713 B1

L1: Entry 6 of 20 File: USPT Nov 26, 2002

US-PAT-NO: 6487713
DOCUMENT-IDENTIFIER: US 6487713 B1

TITLE: Software development system that presents a logical view of project

components, facilitates their selection, and signals missing links prior to compilation

DATE-ISSUED: November 26, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cohen; Frances	Irvine	CA		
Bombet; Marc Abraham	Aliso Viejo	CA		
Lusinsky; Robert Dennis	Placentia	CA		
Lewis; Timothy Andrew	Freemont	CA		
Sandusky; Marc Shane	Aliso Viejo	CA		

US-CL-CURRENT: 717/105

ABSTRACT:

A software development system develops a product from core library of source code elements, the core library being categorized into components having one or more features. A configurator develops configuration state data based on a designated platform type and the source code elements. A graphical user interface displays a visual and logical representation of the product according to the configuration state data, includine visual indications of any unresolved dependencies. A product make routine then generates the product from the source code elements according to the configuration state data.

11 Claims, 45 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 32

☐ 7. Document ID: US 6418424 B1

L1: Entry 7 of 20 File: USPT Jul 9, 2002

US-PAT-NO: 6418424
DOCUMENT-IDENTIFIER: US 6418424 B1

TITLE: Ergonomic man-machine interface incorporating adaptive pattern recognition based control system

DATE-ISSUED: July 9, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven M.	West Harrison	NY	10604	
Hoffberg-Borghesani; Linda I.	Acton	MA	01720	

US-CL-CURRENT: 706/21; 434/178, 706/52

ABSTRACT:

An adaptive interface for a programmable system, for predicting a desired user function, based on user history, as well as machine internal status and context. The apparatus receives an input from the user and other data. A predicted input is presented for confirmation by the user, and the predictive mechanism is updated based on this feedback. Also provided is a pattern recognition system for a multimedia device, wherein a user input is matched to a video stream on a conceptual basis, allowing inexact programming of a multimedia device. The system analyzes a data stream for correspondence with a data pattern for processing and storage. The data stream is subjected to adaptive pattern recognition to extract features of interest to provide a highly compressed representation which may be efficiently processed to determine correspondence. Applications of the interface and system include a VCR, medical device, vehicle control system, audio device, environmental control system, securities trading terminal, and smart house. The system optionally includes an actuator for effecting the environment of operation, allowing closed-loop feedback operation and automated learning.

40 Claims, 31 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 28

☐ 8. Document ID: US 6400996 B1

L1: Entry 8 of 20 File: USPT Jun 4, 2002

US-PAT-NO: 6400996
DOCUMENT-IDENTIFIER: US 6400996 B1

TITLE: Adaptive pattern recognition based control system and method

DATE-ISSUED: June 4, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven M.	West Harrison	NY	10994	
Hoffberg-Borghesani; Linda I.	Acton	MA	01720	

US-CL-CURRENT: 700/83; 370/218, 370/355, 700/17, 700/24, 700/25, 700/86, 700/87, 709/223, 709/227, 715/810, 715/840, 715/841, 718/102, 719/318

ABSTRACT:

An adaptive interface for a programmable system, for predicting a desired user function, based on user history, as well as machine internal status and context. The apparatus receives an input from the user and other data. A predicted input is presented for confirmation by the user, and the predictive mechanism is updated based on this feedback. Also provided is a pattern recognition system for a multimedia device, wherein a user input is matched to a video stream on a conceptual basis, allowing inexact programming of a multimedia device. The system analyzes a data stream for correspondence with a data pattern for processing and storage. The data stream is subjected to adaptive pattern recognition to extract

features of interest to provide a highly compressed representation that may be efficiently processed to determine correspondence. Applications of the interface and system include a video cassette recorder (VCR), medical device, vehicle control system, audio device, environmental control system, securities trading terminal, and smart house. The system optionally includes an actuator for effecting the environment of operation, allowing closed-loop feedback operation and automated learning.

25 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 28

☐ 9. Document ID: US 6199100 B1

L1: Entry 9 of 20 File: USPT Mar 6, 2001

US-PAT-NO: 6199100
DOCUMENT-IDENTIFIER: US 6199100 B1

TITLE: Interactive computer network and method of operation

DATE-ISSUED: March 6, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Filepp; Robert	Springfield	NJ		
Gordon; Michael L.	Dobbs Ferry	NY		
Bidwell; Alexander W.	New York	NY		
Young; Francis C.	Pearl River	NY		
Wolf; Allan M.	Ridgefield	CT		
Meo; Sam	New York	NY		
Tiemann; Duane	Ossining	NY		
Abrahams; Lawrence	Hastings-on-Hudson	NY		
Silfen; Michael J.	Croton-on-Hudson	NY		
Dalsass; Aldo R.	Oakland	NJ		
Lee; Florence M.	Stamford	CT		
Appleman; Kenneth H.	White Plains	NY		

US-CL-CURRENT: 709/203; 719/310

ABSTRACT:

A method is described for operating a distributed processing, interactive computer network. The network is intended to provide very large numbers of simultaneous users access to large numbers of applications which include pre-created, interactive text/graphic sessions. The network includes one or more servers, interactive applications, and one or more user reception systems. The respective reception systems are designed to and are capable of communicating with the respective servers and receiving applications from them. The respective reception systems feature a display interface, as well as reception system software,

operating system software and CPU for executing the applications and presenting applications to respective users. The method features steps for preparing the applications in a high-level programing language so that the applications may be executed at the respective reception systems independently of the reception system CPU type and operating system type by interpreting the respective applications at runtime with an interpreter available at the respective reception systems. In accordance with the method, the applications are divided into sections and structured with objects of multiple types containing application display data and/or program code. The objects are distributed in the network and provided at run time at a reception system at which the respective applications are requested. The method further features steps for providing the respective application programs with a structure that features a header section, data structure section and code section implemented in objects. In this arrangement, method steps are provides to enable respective application programs code section to include one or more procedures for supporting system services, the procedures including a key word for identifying the respective procedures. Additionally, the method includes steps for enabling application program data structure section to identifying the data structure for the respective application programs. Further, the method includes steps for enabling the respective application program structure to include a header section that identifies the respective application program names.

45 Claims, 16 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 16

☐ 10. Document ID: US 6081750 A

L1: Entry 10 of 20 File: USPT Jun 27, 2000

US-PAT-NO: 6081750
DOCUMENT-IDENTIFIER: US 6081750 A

TITLE: Ergonomic man-machine interface incorporating adaptive pattern recognition based control system

DATE-ISSUED: June 27, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven Mark	Yonkers	NY	10701-1705	
Hoffberg-Borghesani; Linda Irene	Acton	MA	01720	

US-CL-CURRENT: 700/17; 345/520, 700/11, 700/56, 700/83, 700/86

ABSTRACT:

An adaptive interface for a programmable system, for predicting a desired user function, based on user history, as well as machine internal status and context. The apparatus receives an input from the user and other data. A predicted input is presented for confirmation by the user, and the predictive mechanism is updated based on this feedback. Also provided is a pattern recognition system for a multimedia device, wherein a user input is matched to a video stream on a

conceptual basis, allowing inexact programming of a multimedia device. The system analyzes a data stream for correspondence with a data pattern for processing and storage. The data stream is subjected to adaptive pattern recognition to extract features of interest to provide a highly compressed representation which may be efficiently processed to determine correspondence. Applications of the interface and system include a VCR, medical device, vehicle control system, audio device, environmental control system, securities trading terminal, and smart house. The system optionally includes an actuator for effecting the environment of operation, allowing closed-loop feedback operation and automated learning.

24 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 28

☐ 11. Document ID: US 6061695 A

L1: Entry 11 of 20 File: USPT May 9, 2000

US-PAT-NO: 6061695
DOCUMENT-IDENTIFIER: US 6061695 A

TITLE: Operating system shell having a windowing graphical user interface with a desktop displayed as a hypertext multimedia document

DATE-ISSUED: May 9, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Slivka; Benjamin W.	Clyde Hill	WA		
Martineau; Teresa Anne	Kirkland	WA		
Brown; Christopher Ralph	Seattle	WA		
Pitt; George	Redmond	WA		
Nakajima; Satoshi	Redmond	WA		
Ramasubtamanian; Sankar	Redmond	WA		
Sheldon; Mike	Redmond	WA		

US-CL-CURRENT: 715/513; 345/629

ABSTRACT:

An operating system shell provides a graphical user interface having a windowing environment with a desktop. The shell synthesizes a hypertext page for display as the desktop in the graphical user interface. The hypertext page has an embedded software object which provides graphical icon-oriented and menu-driven user interface elements for activating operating system services in the displayed hypertext page. The shell also provides windowed hypertext pages for managing file system folders. The shell synthesizes the hypertext pages from templates which can be edited to incorporate a variety of multi-media enhancements with the user interface elements in the graphical user interface. Templates can be associated with specific folders in the file system to provide folder specific hypertext pages integrated with user interface elements for managing the folder.

17 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 12. Document ID: US 5946488 A

L1: Entry 12 of 20 File: USPT Aug 31, 1999

US-PAT-NO: 5946488
DOCUMENT-IDENTIFIER: US 5946488 A
** See image for Certificate of Correction **

TITLE: Method for selectively and incrementally displaying the results of preprocessing

DATE-ISSUED: August 31, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tanguay; David A.	Kitchener			CA
Fraser; Peter J.	Waterloo Region			CA

US-CL-CURRENT: 717/141; 717/127

ABSTRACT:

The present invention makes it possible for computer programmers to selectively examine the effects of preprocessing operations on computer source code. Where a preprocessor construct, such as a macro, appears in the source code, a user may selectively expand the construct to see the effects of expansion. If a macro expands into text which contains other macro calls, the user may selectively expand the other macro calls. The present invention allows the user to go back and forth between unexpanded, partly expanded, and fully expanded constructs in order to obtain a better understanding of the effect of macro expansion on the original source code. The present invention applies to any programming language which supports macros and/or other preprocessing constructs. It also applies to preprocessors that are independent of any programming language, such as the M4 preprocessor. The invention may also be included in another computer program, such as a debugger.

67 Claims, 4 Drawing figures
Exemplary Claim Number: 34
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 13. Document ID: US 5920477 A

L1: Entry 13 of 20 File: USPT Jul 6, 1999

US-PAT-NO: 5920477
DOCUMENT-IDENTIFIER: US 5920477 A

TITLE: Human factored interface incorporating adaptive pattern recognition based controller apparatus

DATE-ISSUED: July 6, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven M.	Yonkers	NY	10701-1705	
Hoffberg-Borghesani; Linda I.	Acton	MA	01720	

US-CL-CURRENT: 382/181; 382/190, 700/83

ABSTRACT:

The need for a more-readily usable interface for programmable devices is widely recognized. The present invention relates to programmable sequencing devices, or, more particularly, the remote controls for consumer electronic devices. The present invention provides an enhanced interface for facilitating human input of a desired control sequence in a programmable device by employing specialized visual feedback. The present invention also relates to a new interface and method of interfacing with a programmable device, which is usable as an interface for a programmable video cassette recorder.

23 Claims, 31 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 27

☐ 14. Document ID: US 5905894 A

L1: Entry 14 of 20 File: USPT May 18, 1999

US-PAT-NO: 5905894
DOCUMENT-IDENTIFIER: US 5905894 A
**** See image for Certificate of Correction ****

TITLE: Meta-programming methods and apparatus

DATE-ISSUED: May 18, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
De Bonet; Jeremy S.	Cambridge	MA		

US-CL-CURRENT: 717/146

ABSTRACT:

The described programming techniques allow for the passing of code arguments to functions, referred to as meta-functions, at compile time through the use of compiler directives. Methods for implementing functions, referred to as meta-loops, which allow for a block of code to be repeated a variable number of times at compile time are also described. The programming methods of the present invention allow for greater code reuse since code arguments can be used to modify the functionality of a meta-function thereby altering its functionality. Meta-functions and meta-loops can also be used to generate a group of functions which share common behaviors. In accordance with the present invention the common behaviors are produced via the use of a common meta-function. This allows the behavior of a group of functions to be altered. This reduces the time and expense associated with creating and maintaining libraries of functions with shared behaviors. Because the programming methods of the present invention can be implemented using compiler commands and functionality supported by existing compiled languages such as C++ they serve to supplement and argument the functionality of existing compiled languages without requiring modifications thereto.

31 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 11

☐ 15. Document ID: US 5901246 A

L1: Entry 15 of 20 File: USPT May 4, 1999

US-PAT-NO: 5901246
DOCUMENT-IDENTIFIER: US 5901246 A

TITLE: Ergonomic man-machine interface incorporating adaptive pattern recognition based control system

DATE-ISSUED: May 4, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven M.	Yonkers	NY	10701-1705	
Hoffberg-Borghesani; Linda I.	Acton	MA	01720	

US-CL-CURRENT: 382/209

ABSTRACT:

An adaptive interface for a programmable system, for predicting a desired user function, based on user history, as well as machine internal status and context. The apparatus receives an input from the user and other data. A predicted input is presented for confirmation by the user, and the predictive mechanism is updated based on this feedback. Also provided is a pattern recognition system for a multimedia device, wherein a user input is matched to a video stream on a conceptual basis, allowing inexact programming of a multimedia device. The system analyzes a data stream for correspondence with a data pattern for processing and storage. The data stream is subjected to adaptive pattern recognition to extract features of interest to provide a highly compressed representation which may be

efficiently processed to determine correspondence. Applications of the interface and system include a VCR, medical device, vehicle control system, audio device, environmental control system, securities trading terminal, and smart house. The system optionally includes an actuator for effecting the environment of operation, allowing closed-loop feedback operation and automated learning.

21 Claims, 31 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 27

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 16. Document ID: US 5883326 A

L1: Entry 16 of 20 File: USPT Mar 16, 1999

US-PAT-NO: 5883326
DOCUMENT-IDENTIFIER: US 5883326 A

TITLE: Music composition

DATE-ISSUED: March 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Goodman; Rodney M.	Altadena	CA		
Spangler; Randall R.	Pasadena	CA		

US-CL-CURRENT: 84/649; 84/634, 84/637, 84/666, 84/669

ABSTRACT:

A music composition system, comprising receiving a first harmony including a first melody, analyzing the first harmony to derive in real-time a rule relating the first melody to the first harmony, receiving a second melody, and applying the rule in real-time to the second melody to produce a second harmony relating to the second melody.

1 Claims, 16 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 14

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 17. Document ID: US 5875108 A

L1: Entry 17 of 20 File: USPT Feb 23, 1999

US-PAT-NO: 5875108
DOCUMENT-IDENTIFIER: US 5875108 A

TITLE: Ergonomic man-machine interface incorporating adaptive pattern recognition based control system

DATE-ISSUED: February 23, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven M.	Yonkers	NY	10701-1705	
Hoffberg-Borghesani; Linda I.	Acton	MA	01720	

US-CL-CURRENT: 700/17; 382/181, 382/190, 700/83

ABSTRACT:

An adaptive interface for a programmable system, for predicting a desired user function, based on user history, as well as machine internal status and context. The apparatus receives an input from the user and other data. A predicted input is presented for confirmation by the user, and the predictive mechanism is updated based on this feedback. Also provided is a pattern recognition system for a multimedia device, wherein a user input is matched to a video stream on a conceptual basis, allowing inexact programming of a multimedia device. The system analyzes a data stream for correspondence with a data pattern for processing and storage. The data stream is subjected to adaptive pattern recognition to extract features of interest to provide a highly compressed representation which may be efficiently processed to determine correspondence. Applications of the interface and system include a VCR, medical device, vehicle control system, audio device, environmental control system, securities trading terminal, and smart house. The system optionally includes an actuator for effecting the environment of operation, allowing closed-loop feedback operation and automated learning.

13 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 28

☐ 18. Document ID: US 5822591 A

L1: Entry 18 of 20 File: USPT Oct 13, 1998

US-PAT-NO: 5822591
DOCUMENT-IDENTIFIER: US 5822591 A

TITLE: Virtual code system

DATE-ISSUED: October 13, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hochmuth; Roland	Fort Colligs	CO		

US-CL-CURRENT: 717/148; 717/164, 719/331

ABSTRACT:

A process for creating new software procedures during run time. An applications program calls a procedure that indirectly corresponds to a set of variables. A library determines that the procedure is not already defined in the library. In a first embodiment, the library builds a source code file to be compiled. In a second embodiment, the library object code creates a string of directives that form part of a command line calling a compiler. The compiler compiles a source code file, selectively compiling only portions of the source code file designated by the command line directives that were created by the library code. For either embodiment, the resulting newly compiled procedure is provided to the applications program by the library in a manner that is transparent to the applications program. No changes are required to the applications program or to the compiler. The library code may add the procedure to the library if there is space, or replace the least recently used or least frequently used procedure if the library is at a capacity limit. As a result, the library can adapt to the set of features being used by the applications program, or the sets of features being used by multiple application programs if the library is shared, or adapt to changes in procedures being used over time.

7 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

☐ 19. Document ID: US 5774357 A

L1: Entry 19 of 20 File: USPT Jun 30, 1998

US-PAT-NO: 5774357
DOCUMENT-IDENTIFIER: US 5774357 A

TITLE: Human factored interface incorporating adaptive pattern recognition based controller apparatus

DATE-ISSUED: June 30, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hoffberg; Steven M.	West Harrison	NY	10604	
Hoffberg-Borghesani; Linda I.	Acton	MA	01720	

US-CL-CURRENT: 713/600; 348/110, 348/27, 348/734, 712/240, 712/245

ABSTRACT:

A human interface device for a user, including a data transmission selector for selecting at least one of a plurality of simultaneously transmitted programs being responsive to an input; a program database containing information relating to at least one the plurality of programs, having an output; a graphical user interface for receiving user commands; and a controller for controlling the graphical user interface and the data transmission selector, the controller determining a user characteristic, receiving the output of the program database and presenting, based

on the user characteristic and the program database, information relating to at least one of the plurality of programs on the graphic user interface in association with a command, the graphic user interface allowing the user to select the command and thereby authorize an operation in relation to the at least one of the plurality of programs. An objective user characteristic is detected based on one or more temporal-spatial user characteristics of the input, including a velocity component, an efficiency of input, an accuracy of input, an interruption of input and a high frequency component of the input signal.

27 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 28

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 20. Document ID: US 5513305 A

L1: Entry 20 of 20 File: USPT Apr 30, 1996

US-PAT-NO: 5513305
DOCUMENT-IDENTIFIER: US 5513305 A

TITLE: System and method for documenting and displaying computer program code

DATE-ISSUED: April 30, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Maghbouleh; Arman	Stanford	CA		

US-CL-CURRENT: 715/500; 715/781

ABSTRACT:

A system and method for documenting and displaying computer program code comprises a token annotation unit, a comment analyzer, a token parsing library, and a code outline unit. The token parsing library parses a program comprising related program code files into a set of constituent tokens. The token annotation unit selectively associates one or more annotations with tokens in a program by creating a token annotation object. When program code is displayed, the token annotation unit visually identifies each annotated token present according to a display style. The token annotation unit presents annotations corresponding to an annotated token in response to the selection of the annotated token during program code display.

20 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 13

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
PREPROCESSOR AND DIRECTIVE AND COLOR	20

Display Format: REV Change Format

Previous Page Next Page Go to Doc#

Google Web Images Groups News Froogle more »

preprocessor directive color Search Advanced Search Preferences

Web Results 1 - 10 of about 11,100 for preprocessor directive color. (0.29 seconds)

1. The Code Project - A C++ to HTML conversion utility - C++ / MFC
... style rules for comments, keywords and **preprocessor directives**. So, you don't have to do a search-and-replace if you want to change, say, the **color** of keywords ...
www.codeproject.com/cpp/cpphtml.asp - 41k - Cached - Similar pages

on screen example highlighted words. Operator >>
12/3/04 TT

2. Ada-ASSURED Reference Manual
... Care is needed when using **preprocessor directives** in lists ... in a list is a **preprocessor** construct, then ... Use_Green: constant BOOLEAN := TRUE; type **Color** is (Red ...
www.grammatech.com/aadoc/man-**preprocessor**.html - 8k - Cached - Similar pages

3. C Within Help File
... is first passed through aC/C++ **preprocessor**, you will ... include line An #include line contains an #include **directive**. ... Its background **color** is: A single red space ...
www.thinkage.ca/english/products/cwithin-helpfile.html - 20k - Cached - Similar pages

Pascal User's Guide: A - Pascal **Preprocessor**
... The cppas **preprocessor** handles the Pascal conditional variables and compiler **directives**. ... red elseif.p hostname% a.out The **color** is red ... The %elseifdef **Directive**. ...
web.mit.edu/sunsoft_v5.1/www/ pascal/user_guide/pascalugApcpp.doc.html - 43k - Cached - Similar pages

[PDF] Object Orientated Design and C++ Objectives Books Introduction to ...
File Format: PDF/Adobe Acrobat - View as HTML
... Example of OOD: Shapes • Base Object • Derived Objects Shape: common data • centre • colour common operations • location • translation ...
www.ee.surrey.ac.uk/Personal/ R.Bowden/cpp/notes/handouts.pdf - Similar pages

C++ Coding Conventions Examples
... we could specify: c=green; to set the colour. ... Include **directives**, #include "Molecule.h", the quotes indicate a ... will define and test a **preprocessor** variable ...
people.cryst.bbk.ac.uk/~classlib/bioinf/example.html - 14k - Cached - Similar pages

3 : Ways of specifying policy values
... A number of policies have equivalent **preprocessor directives**. ... **Directive** keyword(s), Related policies. TABLE_BGCOLOR, Default TABLE colour. ...
rcum.uni-mb.si/local/a2h/policy_manual_3.html - 15k - Cached - Similar pages

7 : Using the **preprocessor**
... Note_1: Strictly speaking the "use **preprocessor**" line above isn't needed as this is set to "yes ... **Directive** Value, Effect. TABLE_BGCOLOR Colour, Colour of background ...
rcum.uni-mb.si/local/a2h/a2hdoco_7.html - 28k - Cached - Similar pages
[More results from rcum.uni-mb.si]

4. How to use c2html
... account. (remaining arguments are optional) [**directive color**] is the **color** of #includes and other **preprocessor directives**. [keyword ...
wwwcgi.umr.edu/cgi-bin/cgiwrap/schuette/c2html.new.pl - 2k - Cached - Similar pages

3 : Ways of specifying policy values

... SIMPLE, Keep it simple. 3.4 Changing policies by using **preprocessor directives**.
A number of policies have equivalent **preprocessor directives**. ...
wwwvms.mppmu.mpg.de/vmsdoc/ asctohtm/policy_manual_3.html - 22k - Cached - Similar pages

Gooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

preprocessor directive color Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

C++ (VC7.1, VC7, VC6)
Windows (WinXP, Win2K, Win2003, Win95, Win98, STL, Win32, VS Dev
Posted 25 May 2004
Articles by this author
6,716 views

A C++ to HTML conversion utility

By q123456789

A utility which converts your C++ code to HTML.

8 members have rated this article. Result:
Popularity: 3.85. Rating: **4.**

Download source and exe files - 69.1 Kb

Introduction

Cpphtml is a utility to convert your C++ code to HTML. If you have a C++ file, *myprogram.cpp*, and you want to put it on your website, you can run it through which will convert the code to HTML with all comments, keywords and preproce directives highlighted. *Cpphtml* will send all output to `cout`, so you have to redi output to a file if you want to create a HTML file:

```
C:\>cpphtml myprogram.cpp >myprogram.htm
```

Cpphtml will convert all tabs to 4 spaces. If you want the tab size to be 8 space specify the tab size on the command line:

```
C:\>cpphtml myprogram.cpp 8 >myprogram.htm
```

The HTML code contains a `<style>` element which contains style rules for comr keywords and preprocessor directives. So, you don't have to do a search-and-r you want to change, say, the color of keywords. For example, if you want all ke bold, just change the `.keyword` style rule: `.keyword{color:rgb(0,0,255);for weight:bold}`. It's that easy.

I don't claim *Cpphtml* works perfectly. I tested it on the Dinkumware STL files, of *Cpphtml*, and a large Microsoft CPP file. The results are great. *Cpphtml* was c with the Borland C++ 5.5 command line compiler: *bcc32 cpphtml.cpp*.

A walk through the code

```
#include<fstream>
```

```
#include<string>
#include<ctype.h>
```

Cpphtml will replace all tabs by 4 spaces if no tab size is specified. Change `_TAI` if you want the default tab size to be 8.

```
#define _TABSIZE    4

using namespace std;

int tabsize = _TABSIZE;
```

Token is a class which represents chunks of code. A token can have the type *cc pp (preprocessor directive)*, *keyword* or *code*. *Code* is everything which is not a *comment*, *pp* or *keyword*. Note that there are no getter and setter methods: be `operator>>` and `operator<<` are friends of class token, we don't need any.

```
class token {
public:
    token() : _what(code) {}
protected:
    enum type {code, comment, pp, keyword};
    string _str;
    type _what;
    friend istream& operator>>(istream&, token&);
    friend ostream& operator<<(ostream&, const token&);
};
```

The function `iskeyword()` returns `true` if string `s` is a C++ keyword, `false` if n possible you don't recognize some keywords, e.g. `and`. Those keywords can be programmers who don't have access to all ASCII characters. I've never seen co such keywords though.

```
bool iskeyword(const string& s)
{
    static const char* keywords[] = {
        "and",
        "and_eq",
        "asm",
        "auto",
        "bitand",
        "bitor",
        "bool",
        "break",
        "case",
        "catch",
        "char",
        "class",
        "compl",
        "const",
        "const_cast",
        "continue",
        "default",
        "delete",
        "do",
        "double",
```

```
            "dynamic_cast",
            "else",
            "enum",
            "explicit",
            "export",
            "extern",
            "false",
            "float",
            "for",
            "friend",
            "goto",
            "if",
            "inline",
            "int",
            "long",
            "mutable",
            "namespace",
            "new",
            "not",
            "not_eq",
            "operator",
            "or",
            "or_eq",
            "private",
            "protected",
            "public",
            "register",
            "reinterpret_cast",
            "return",
            "short",
            "signed",
            "sizeof",
            "static",
            "static_cast",
            "struct",
            "switch",
            "template",
            "this",
            "throw",
            "true",
            "try",
            "typedef",
            "typeid",
            "typename",
            "union",
            "unsigned",
            "using",
            "virtual",
            "void",
            "volatile",
            "wchar_t",
            "while",
            "xor",
            "xor_eq"
        };

        for (int i = 0; i < sizeof(keywords) / sizeof(char*); i++)
            if (string(keywords[i]) == s)
                return true;

        return false;
    }
```

The function `containspp()` returns `true` if string `s` contains a substring which i

preprocessor directive. A token of type *pp* can contain a string of the form "#.. therefore, we have to find a substring.

```
bool containspp(const string& s)
{
    static const char* pptokens[] = {
        "define",
        "elif",
        "else",
        "endif",
        "error",
        "if",
        "ifdef",
        "ifndef",
        "include",
        "line",
        "pragma",
        "undef"
    };

    for (int i = 0; i < sizeof(pptokens) / sizeof(char*); i++)
        if (s.find(pptokens[i]) != string::npos)
            return true;

    return false;
}
```

`Operator>>` extracts a token from an input stream. It recognizes "//" and "/*. comments, preprocessor directives of the form "#...define", and keywords. S constants are also recognized to avoid keywords to be highlighted in strings.

```
istream& operator>>(istream& is, token& t)
{
    t._str = "", t._what = token::code;
    int c = is.get();
    switch (c) {
        case '/':
            c = is.get();
            if (c == '*') {
                t._str = "/*";
                t._what = token::comment;
                while (1) {
                    c = is.get();
                    if (c == EOF)
                        return is.unget(), is.clear(), is;
                    if (c == '/') {
                        if (t._str.length() > 2 &&
                            t._str[t._str.length() - 1] == '*') {
                            return t._str += '/', is;
                        }
                    }
                    t._str += (char)c;
                }
            } else if (c == '/') {
                t._str = "//";
                t._what = token::comment;
                c = is.get();
                while (c != '\n' && c != EOF) {
                    t._str += (char)c;
                    c = is.get();
                }
```

```
            if (c == '\n') {
                t._str += '\n';
            }
            return is;
        }
        t._str = '/';
        return is.unget(), is.clear(), is;
    case '#':
        t._str = '#';
        c = is.get();
        while (strchr(" \r\n\t", c)) {
            t._str += (char)c;
            c = is.get();
        }
        if (c == EOF)
            return is.unget(), is.clear(), is;
        while (strchr("abcdefghijklmnopqrstuvwxyz", c)) {
            t._str += (char)c;
            c = is.get();
        }
        is.unget(), is.clear();
        if (containspp(t._str))
            t._what = token::pp;
        return is;
    case '\'':
    case '"': {
        char q = (char)c;
        t._str = q;
        while (1) {
            c = is.get();
            if (c == EOF)
                return is.unget(), is.clear(), is;
            if (c == q) {
                if (t._str.length() >= 2) {
                    if (!(t._str[t._str.length() - 1] == '\\' &&
                        t._str[t._str.length() - 2] != '\\'))
                        return t._str += q, is;
                } else {
                    return t._str += q, is;
                }
            }
            t._str += (char)c;
        }
    }
    case 'a':
    case 'b':
    case 'c':
    case 'd':
    case 'e':
    case 'f':
    case 'g':
    case 'i':
    case 'l':
    case 'm':
    case 'n':
    case 'o':
    case 'p':
    case 'r':
    case 's':
    case 't':
    case 'u':
    case 'v':
    case 'w':
    case 'x':
        t._str += (char)c;
```

```
            c = is.get();
            while (isalpha(c) || isdigit(c) || c == '_') {
                t._str += (char)c;
                c = is.get();
            }
            is.unget(), is.clear();
            if (iskeyword(t._str))
                t._what = token::keyword;
            return is;
        case EOF:
            return is;
        default:
            t._str += (char)c;
            c = is.get();
            while (c != '/' && c != '#' && !strchr("abcdefgilmnoprstuvwx"
                c != '\'' && c != '"' && c != EOF) {
                t._str += (char)c;
                c = is.get();
            }
            is.unget(), is.clear();
            return is;
    }
}
```

The function `html()` replaces the characters '&', '<', '>' and '"' in string `s` by its equivalents and replaces the tabs by spaces.

```
string html(const string& s)
{
    string s1;
    string::size_type i;
    for (i = 0; i < s.length(); i++) {
        switch (s[i]) {
            case '&':
                s1 += "&";
                break;
            case '<':
                s1 += "<";
                break;
            case '>':
                s1 += ">";
                break;
            case '"':
                s1 += """;
                break;
            case '\t':
                s1.append(tabsize, ' ');
                break;
            default:
                s1 += s[i];
        }
    }
    return s1;
}
```

`Operator<<` sends a token to an output stream. The code is straightforward.

```
ostream& operator<<(ostream& os, const token& t)
{
    if (t._what == token::code)
```

```
        cout << html(t._str);
    else if (t._what == token::comment)
        cout << "<span class=comment>" << html(t._str) << "</span>";
    else if (t._what == token::keyword)
        cout << "<span class=keyword>" << html(t._str) << "</span>";
    else if (t._what == token::pp)
        cout << "<span class=pp>" << html(t._str) << "</span>";
    else
        cout << html(t._str);
    return os;
}
```

This is the entry point of *Cpphtml*. All code will be wrapped in a `<pre>` element. overloading `operator>>` and `operator<<`, the `while` loop is very short and clea output is sent to `cout`.

```
int main(int argc, char **argv)
{
    if (argc != 2 && argc != 3) {
        cout << "usage: cpphtml file [tab size]" << endl;
        return 0;
    }
    ifstream is(argv[1]);
    if (!is.good()) {
        cerr << "bad input file" << endl;
        return -1;
    }
    if (argc == 3) {
        tabsize = atoi(argv[2]);
        if (tabsize <= 0)
            tabsize = _TABSIZE;
    }
    cout << "<html>" << endl
      << "<head>" << endl
      << "<style>" << endl;
    cout << ".keyword{color:rgb(0,0,255);}" << endl;
    cout << ".comment{color:rgb(0,128,0);}" << endl;
    cout << ".pp{color:rgb(0,0,255);}" << endl;
    cout << "</style>" << endl << "<body>" << endl;
    cout << "<pre style=\"font-family:courier;font-size:10pt\">";
    token t;
    while (is >> t) {
        cout << t;
    }
    cout << "</pre>" << "</body>"
         << endl << "</html>" << endl;
    return 0;
}
```

q123456789

Click here to view q123456789's online profile.

Other popular articles:

- Your fractal, Sir. -Thank you, James.
  The Fractal Template Library
- A simple software key useful to protect software components

This article shows a way to implement a base software key that could be useful for protectir components.

- How a C++ compiler implements exception handling
  An indepth discussion of how VC++ implements exception handling. Source code includes e handling library for VC++.
- Scaling of memory intensive multi-threaded applications to SMMP comput
  This article discusses impact of the multithreaded run-time library's memory manager on sc memory-intensive server applications to Shared Memory Multiprocessor computers.

[Top] **Sign in** to vote for this article: *Poor* ○ ○ ○ ○ ○ *Excelle*

FAQ Noise level Medium Search comments Set Options

View Message View Per page 25

New thread Msgs 1 to 1 of 1 (Total: 1) (Refresh) First Prev Next Last

Subject	Author	Date
Thanks, It's very good	**lynhoo**	**21:49 2 Jun '04**

Last Visit: 13:37 Friday 3rd December, 2004 First Prev Next Last

All Topics, MFC / C++ >> C++ / MFC >> Utilities
Updated: 25 May 2004

Using a Preprocessor


| Online Documentation Home Page

2. Using a Preprocessor

This chapter describes the use of **Ada-ASSURED** when code is written for preprocessing. Support for preprocessors involves the optional extension of the Ada language with two constructs discussed below in:

- Section Preprocessor Directives
- Section Dollar-prefixed Identifiers

Omit this chapter if you do not use a preprocessor.

Preprocessor Directives

Preprocessor directives start with a #-sign at the beginning of a line and include all the text up to the end of the line. Because the text after column one is uninterpreted, all preprocessors that use the column-one #-sign syntax are supported. In particular, Rational's VADS compiler, which is also SunPro's Sun Ada compiler, is supported.

The availability of this extended syntax is optional. The resource **preprocessorOk** controls whether the syntax of preprocessor directives is accepted or rejected as a syntax error.

When a program is formatted, all preprocessor directives are displayed in a distinctive style so they are easily distinguished.

Preprocessor directives are only permitted in lists of the following syntactic categories:

- **association**,
- **compilation_unit**,
- **component**,
- **component_association**,
- **component_clause**,
- **context**,
- **declaration**,
- **entry_item**,
- **enumeration_literal_specification**,
- **exception_handler**,
- **generic_parameter**,
- **parameter_specification**,

- **pragma**,
- **protected_operation_item**, and
- **statement**.

This disciplined use of preprocessor directives makes programs more readable. For example, the following conditional compilation of the left-hand-side of an assignment statement

```
          X :=
#if Sparc then
                Y
#else
                Z
#end if;
          ;
```

must be written as

```
#if Sparc then
          X := Y;
#else
          X := Z;
#end if;
```

The # template command appears in the context pane whenever a preprocessor directive may be inserted. When a preprocessor directive is typed at an indented placeholder, the #-sign need not be typed in the first column of the window—it is sufficient that the #-sign be the first character entered. The formatter will move the text to the first column, as necessary.

Care is needed when using preprocessor directives in lists that require separators (e.g. parameter_specification) as opposed to lists that use terminators (e.g. statement). If the last item in a list is a preprocessor construct, then an extra separator may be inserted. For example, the following enumeration declaration will result in an extra separator:

```
#Use_Green: constant BOOLEAN := TRUE;
      type Color is
                (Red,
                 Blue,
#if Use_Green then
                 Green,
#end if;
                 );
```

This problem can be avoided by putting the conditional parts of the declaration in the middle, as in:

```
#Use_Green: constant BOOLEAN := TRUE;
      type Color is
                (Red,
#if Use_Green then
                 Green,
#end if;
                 Blue
                 );
```

Dollar-prefixed Identifiers

A dollar-prefixed identifier is an Ada identifier that begins with a $.

The availability of this extended syntax is optional. The resource **dollarIdentifiers** control whether the syntax of dollar-prefix identifiers is accepted or rejected as a syntax error.

When a program is formatted, all dollar-prefixed identifiers are displayed in a distinctive style so they are easily distinguished.

Dollar-prefixed identifiers are permitted wherever a name or identifier is allowed.

C Within Help File

This page recreates the current Help file for **C Within**.

---

3. Introducing C Within

C Within lets you inspect the operations performed on a text file by the C++ or C preprocessor. Usually the text file is a C++ or C source file. The purpose of **C Within** is to help you find bugs resulting from complicated uses of the C/C++ preprocessor.

C Within lets you see how preprocessor macros are expanded, from the original source, through all the intermediate macro expansions, to the final preprocessed source. You can interactively expand or collapse each level of macro expansion, for each individual macro.

C Within also lets you step into **#include** files. Dead code (resulting from inactive **#if** branches) is lowlighted. Together, these allow you to easily see the flow of preprocessing.

Note: Thinkage Ltd. has obtained U.S. patent #5,946,488 for the processing algorithms that allow **C Within** to expand and unexpand preprocessor directives.

Other Topics

General Concepts of C Within

Phases of Operation

C Within first runs an enhanced C/C++ preprocessor on your text file. If this phase is successful, you can then interactively browse through the file, inspecting particular macro expansions and **#include** files.

The browser presents you with a split window. The top window contains any diagnostic messages generated by the preprocessing phase. This will include any warnings about suspicious or non-portable preprocessing actions. It will also include any errors encountered during preprocessing, such as the inability to find particular **#include** files.

If no preprocessing errors occur, the bottom right window presents an interactive view of your source.

You can increase your viewing area by dragging the dividing bar to minimize the preprocessor report window.

The bottom left window is a tree view of the **#include** structure of your text. You can use this to jump directly to a particular instance of an **#include**d source text, without having to search for all the relevant **#include** statements in the parent texts.

Configuration Files

Since the input file for **C Within** is first passed through a C/C++ preprocessor, you will usually want to specify things such as the **#include** search orders and additional macro definitions (such as would appear on compiler command lines, rather than in the source files). You can also tell the browser whether you are using C or C++. This affects how **C Within** treats constructs like // comments and tokens that are only supported by C++.

You specify these things via a typical Windows dialogue. Since you will frequently want to use the same settings, you can save them in a configuration file and reload them later.

Browsing

After a successful preprocessing, the bottom right window will display an interactive view of your input file. Elements are color-coded to make them easier to identify. Colors can be changed by using the View menu, but the default colors are as follows:

Dead code
: Dead code is text that will not be compiled because of **#if** directives. By default, dead code is shown in light gray (although the View menu offers an option that can make dead code disappear entirely).

`#include line`
: An **#include** line contains an **#include** directive. Such lines are displayed in red (unless they are in dead code).

Macro identifier
: A macro identifier is a macro name that would be expanded if the code is compiled. It is shown with a blue or green background.

Expansion text
: Expansion text is text that is the result of macro expansion. Its background color is:
  - A single red space if the macro expands to nothing. This gives you something to click on if you want to unexpand the macro again.
  - Green if the text contains a macro that can be expanded further.
  - Yellow otherwise.

Along the left edge of the view of the input file, you will see a red arrow referred to as the **marker**. The status bar will show the file and line number of the marker. You can change the location of the marker by left clicking along the left edge of the view.

Right-clicking any text displays a context menu specifying the operations that can be performed on that text. The following operations may be available on the context menu (depending on the text being clicked):

`Expand`

Expands a macro into its expanded text. This action is only available when the text is an expandable macro identifier.

`Unexpand`
: Collapses the expansion of a macro back into the original macro expression (the macro identifier and its arguments). This action is only available when the text is a macro expansion.

`Goto Definition`
: Goes to the **#define** directive that defined a particular macro. This action is only available when the text is a macro identifier. If the **#define** directive was given in another file, **C Within** goes to that file. The **#define** directive is highlighted with an arrow to make it easier to identify on the screen.

`Expand Line`
: Fully expands all macros on the line.

`Unexpand Line`
: Fully collapses all macros on the line.

`Expand to Marker`
: Performs `Expand Line` on all lines from the marker to the mouse pointer.

`Unexpand to Marker`
: Performs `Unexpand Line` on all lines from the marker to the mouse pointer.

`Open Include`
: Opens an **#included** file for browsing. This action is only available when the text is an **#include** directive.

`Previous Marker`
: Returns to the previous file and position within that file (if any). This action is available anywhere, provided that you have looked at a file previously in this session.

`Copy to Clipboard`
: Copies all lines from the marker to the current mouse position, and places the text in the Windows clipboard. The lines are copied as text, using the currently presented level of macro expansion.

Instead of selecting these actions from the context menu, you can get the same effect from mouse shortcuts.

- Clicking the middle and left mouse buttons simultaneously on any text is equivalent to `Previous Marker`.
- Clicking the left mouse button on an expandable macro expands the macro. Clicking the left mouse button on an **#include** statement is equivalent to `Open Include`.
- Clicking the middle mouse button on a macro expansion is equivalent to `Unexpand`.

Note that actions involving the middle button are only available if you have a three-button mouse. Also note that on Windows 95, the default meaning of the middle button is a double-click of the left button. To use the middle mouse button as described above, you must use the Windows 95 control panel to prevent Windows 95 from changing a middle-click to a left double-click; you can do this by making a middle-click "undefined". (This tells Windows 95 not to convert a middle-click into something else.)

The File Menu

This menu lets you select which files to view.

`Open`
: Selects a file to view via a standard Windows file open dialogue. You can only select one file at a time.

`Reopen`
: Reopens the currently viewed file. This is needed if the file has been modified during your viewing session.

Most recently used files
: Opens a previously viewed file. Warning: this does *not* load the configuration that was in effect at the time that file was viewed; it uses the configuration that is currently set.

`Exit`
: Terminates C Within.

The Configuration Menu

This menu lets you specify the preprocessing parameters.

`Set`
: Presents a dialogue to set **#defines**, **#include** search paths, and the recognition of constructs that are by C++ but not C. The macro body of a define follows the macro identifier, as it does in a **#define**.

`Load`
: Loads a previously created configuration. This file becomes the current configuration file.

`Load From Project`
: Loads configuration parameters from an IDE project file. Currently, only Microsoft Developer Studio 5.0 projects are supported. The parameters are loaded as for the first source file of the first target in the project file.

`Save`
: Saves the current configuration into the current configuration file.

`Save As`
: Saves the current configuration into a new file. This file becomes the current configuration file.

The View Menu

This menu lets you control the appearance of the browsing view.

`Status Bar`
: Shows or hides the status bar. The status bar shows the name of the file being viewed, as well as some brief usage comments.

`Set Tab Stops`
: Sets the number of characters between tab stop positions.

`Set Font`
: Sets the font used to view the preprocessed input file.

`Set Color`
: Set the colors used to present the preprocessed input file.

`Show Dead Code`
: Dead code is code that will not be compiled because of **#if** and similar constructions. If the `Show Dead Code` option is turned on, **C Within** displays such code in light gray. If `Show Dead Code` is turned off, dead code will not be shown at all; **C Within** only shows the code that will actually be compiled.

`Find Marker`
: Moves the view so that it includes the marker. The marker will be at the top of the view, except near the end of the subfile.

`Search`

Searches for text in the current subfile. The search runs forward from the marker until it finds a match. At the end of the file, the search wraps around to the beginning of the file; if no match is found, it stops at the original marker position. The text searched is only the text that is currently seen—it does not search into unexpanded macros.

The following characters have special meanings in the search patterns:

`^`
: matches start of line

`$`
: matches end of line

`.`
: matches any character

`*`
: matches zero or more of the previous character or closure

`+`
: matches one or more of the previous character or closure

`?`
: matches zero or one of the previous character or closure

`[`
: starts a character closure (as in `[abc]`)

`]`
: ends a character closure

`^`
: can be used as the first character in a closure to indicate all characters not in the closure; `[^AB]` means all characters except `A` or `B`

`Reverse Search`

Same as `Search`, but searches in the opposite direction.

`Goto Line`

Moves the marker to the next occurrence of the given line number in the current subfile. The search starts from the current marker and wraps around at the end of the file, stopping at the original marker position if no match is found. (Note that there may be several lines with the same number because of **#line** directives.)

The Help Menu

This menu presents information about **C Within**.

`Help Topics`

Shows documentation on the use of **C Within**.

`About C Within`

Shows copyright notice and contact information for Thinkage Ltd.

Command Line Parameters

The general command line format is:

`cwithin` *`options file`*

This opens the given file for browsing.

Possible options are:

`/Define:`*`macro`*
: Defines a macro, as if a **#define** directive was inserted at the beginning of the text to be preprocessed. The syntax of the definition is the same as in a normal **#define** directive with the word **#define** omitted. If there are spaces in the macro definition, the whole string should be enclosed in quotes. For example, `/Define:"MYMACRO 3"` has the same effect as

```
#define MYMACRO 3
```

`/user_Include:`*`directory`*
: Adds the given directory to the search path for **#include** directives.

`/System_include:`*`directory`*
: Adds the given directory to the search path for **#include** directives. This only affects **#include** directives that enclose the filename in angle brackets (for example, **#include <file.h>**).

`/Project:`*`projfile`*
: Opens an IDE project file specified by the *`projfile`* pathname. **C Within** tries to determine what kind of project file it is, then reads the file to determine which macros should be predefined and what #include search paths should be used. The project file is opened in a manner similar to the `Load from Project` operation on the Configuration menu. This option may only be specified once on the command line.

`/C++-`
: Tells the preprocessor not to run in C++ mode. In other words, you are looking at C code rather than C++. **C Within** will not recognize C++ comments or other special constructs. You may only specify this option once on the command line.

`/C+++`
: Explicitly tells the preprocessor to run in C++ mode. This is only necessary if you want to override settings in a project file that say the source code is C only (not C++). You may only specify this option once on the command line.

`/HELP`
: Displays a message box showing the syntax of the **C Within** command line. If you specify this option, **C Within** does not attempt to do any preprocessing or show any processed output.

Keywords in the above options can be abbreviated by omitting any underscores and/or letters shown in lower case. For example, `/Project` may be abbreviated as `/proj`, `/pj`, `/p` and so on. Keywords may also be typed in upper case, lower case, or mixed case.

Installing C Within as a Microsoft Developer Studio Tool

If you use Microsoft Developer Studio 5.0, you will want to install **C Within** as a tool that is known to the studio. This will make it easier for you to invoke **C Within** on the source files of your programming projects.

1. From the **Tools** menu, select the **Customize** item, then select the **Tools** property page.
2. In the Menu box, select the **Open** entry and set the tool name (for example, C Within).
3. In the Command box, set the pathname of **C Within** (wherever you installed it).
4. In the Arguments box, insert the following:

```
"/project:$(WkspDir)\$(WkspName).dsp" "$(FilePath)"
```

5. In the Initial directory box, insert `$(FileDir)`.

To launch the browser, first open a file for editing in the Developer Studio. With that editing window selected, bring up the **Tools** menu and select the C Within item (or whatever name you entered in Step 2 above).

Registering C Within

To register your copy, please fill out the registration form in the file `register.txt` (found in the same folder where you installed **C Within**). The file provides complete information on how to register the software with Thinkage. In return, Thinkage will supply a registration number that you can use to register your copy of the software.

Once you have received a registration number from Thinkage, start **C Within** and select the **Register** item of the **File** menu. Type your registration number into the input boxes, and your copy of **C Within** will be registered.

How to use c2html

The correct usage of c2html is as follows:

A. `<path_to_script>/c2html.pl?<filename>,[directive color],[keyword color],[comment col`

`<path_to_script>` is the path to the script. For UMR students, this is `http://wwwcgi.umr.edu/cgi-bin/cgiwrap/<username>` assuming that the script was copied to your account.
(remaining arguments are optional)

`[directive color]` is the color of `#includes` and other preprocessor directives.
`[keyword color]` is the color of the various keywords such as `if, int, new`,etc.
`[comment color]` is the color of all comment blocks.
`[bgcolor]` is the background color of the page to be generated.
`[text color]` is the color of the text to be generated.

Here is an example:
`http://wwwcgi.umr.edu/cgi-bin/cgiwrap/schuette/c2html.pl?queue.cpp,,,,000000,FFFFFF`
This will display the file `queue.cpp` with a black background and white text. The remaining colors use the defaults.

c2html created by Matt Schuette and Dave Batton.